Vigilant Medical

ImageShare

Security Protocols, Risk Assessment, and Risk Management for HIPAA-Covered PHI

Rev 1.2   JUNE 2011

Previous Revisions:

Revision	Date
Initial (1.0)	January 2010
Database Update (1.1)	August 2010

Overview

The Vigilant ImageShare Application utilizes web-standards compliant security methods to secure Protected Health Information [PHI] moving through the system ("in-flight" data) and stored in the system ("at-rest" data). This document outlines the security protocols implemented on this system, including methods by which clients may verify active security protocols. This document outlines known risks and the measures implemented to mitigate them. This document will reference and show the implementation of protections specific to hosted cloud solutions and provided by the Amazon Web Services HIPAA-compliance guidance document[1].

Security Aim

The Vigilant ImageShare system is implemented to ensure the security and the integrity of PHI. The security protocols used in the ImageShare system must meet or exceed the industry-standard practice for the transfer and storage of PHI. Moreover, the protocols must meet or exceed the guidelines for covered entities under the Health Insurance Portability and Accountability Act of 1996[2]. To this end, the Privacy and Security Standards outlined by the Department of Health and Human Services [DHHS][3] and the associated X12N HIPAA implementation interpretations shall be regarded as the minimum requirements for ImageShare and Vigilant's interactions with PHI.

The security of the ImageShare service is vastly improved by the cooperation and communication between Vigilant technical staff and the information technology personnel of the subscribing institution. To that end, this document provides thorough coverage of Vigilant's security principles and protocols. Security policies vary between institutions and no document can provide indications for every possible misuse scenario. To address this, questions and clarifications will be provided by the Vigilant technical staff on request by the subscribing institution. Vigilant will maintain communication channels within the institutional information technology group and make them immediately aware of any issues or changes to this document. As part of this communication effort, Vigilant expects the subscribing institution to make Vigilant immediately aware of any suspected problems or deficiencies in this document so that they may be addressed.

[1] Creating HIPAA-Compliant Medical Data Applications with Amazon Web Services, April 2009, Amazon Web Services: http://media.amazonwebservices.com/AWS_HIPAA_Whitepaper_Final.pdf

[2] Health Insurance Portability and Accountability Act of 1996, Public Law 104-191, 104th Congress: http://www.cms.gov/HIPAAGenInfo/Downloads/HIPAALaw.pdf

[3] Privacy and Security Standards HIPAA, Department of Health and Human Services: http://www.cms.gov/HIPAAGenInfo/04_PrivacyandSecurityStandards.asp

System Design

This concise list enumerates the largest components of each layer. The security of the entire system is highly dependent upon the security of its constituent parts. All components utilized in ImageShare are mature, secure, and widely used. Questions about the security of any individual component should be directed to the original provider. Links to the component security information are provided when available.

Physical Layer

Amazon Web Services Elastic Compute Cloud[4]
Amazon Web Services Elastic Block Storage

Data Layer

Microsoft SQL Server 2008[5]
Microsoft New Technology File System

Application Layer

Microsoft Windows Server 2008: Datacenter
Microsoft .NET Framework 4.0
Microsoft ASP.NET 4.0[6]
Microsoft ASP .NET MVC2 Framework

Transport Layer

Microsoft Internet Information Services 7.0[7]

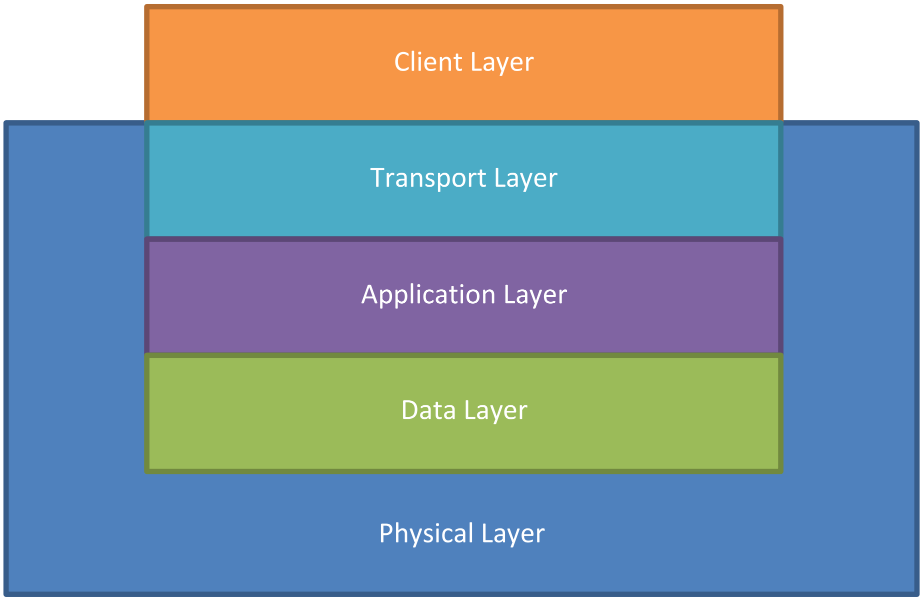
Client Layer

Oracle (Formerly Sun) Java 6.0 Update 27[8]
Microsoft Internet Explorer 7[9], 8[10]
Mozilla Firefox 3.0+[11]
Google Chrome 5+[12]

[4] AWS Security Center, Amazon Web Services: http://aws.amazon.com/security
[5] Overview of Security Features in SQL Server 2008, Microsoft: http://technet.microsoft.com/en-us/magazine/2008.04.sqlsecurity.aspx
[6] ASP.NET Web Application Security, Microsoft: http://msdn.microsoft.com/en-us/library/330a99hc.aspx
[7] Enhanced Server Protection: Overview, Microsoft: http://www.iis.net/overview/EnhancedServerProtection
[8] Java SE 6 Documentation, Oracle: http://download.oracle.com/javase/6/docs/index.html
[9] Internet Explorer 7 Desktop Security Guide, Microsoft: http://www.microsoft.com/downloads/en/details.aspx?FamilyID=6aa4c1da-6021-468e-a8cf-af4afe4c84b2&displaylang=en
[10] Internet Explorer 8 Desktop Security Guide, Microsoft: http://www.microsoft.com/downloads/en/details.aspx?displaylang=en&familyid=44405777-51b4-4376-9cef-f0341b13fcde
[11] Firefox web browser | Features…, Mozilla: http://www.mozilla.com/en-US/firefox/features/#security
[12] Google Chrome – Browser Security, Google: http://www.google.com/chrome/intl/en/more/security.html

Exposures

This diagram shows all of the external exposures through which data may be accessed on the ImageShare system:



The external faces of this diagram show the exhaustive list of security risks to the ImageShare system. These risks are common to all web applications. For instance, the data layer is never directly vulnerable. Any attempt to access the data layer must come through the client, transport, and application layers, or through physical access to the server.

For data "at-rest," the security risks are, therefore, the union of risks to the intersection of risks to the physical and data layers with the intersection of risks to the Client, Transport, Application, and Data layers. More concisely:

$$\begin{gathered}Let\ the\ security\ risk\ to\ any\ particular\ layer\ be\\ denoted\ by\ the\ two\ letter\ acronym\ for\ that\ layer.\\ Then,\end{gathered}$$

$$Total\ DL\ risk = (PL \cap DL) \cup (CL \cap TL \cap AL \cap DL)$$

Noting that the former grouping is defined by physical risks and that the second grouping is defined by software risks, there will be a nearly empty intersection of risks among the groupings and that the unit analysis of risks for each layer is sufficient to describe the risk of the entire system.

Security Principles

Vigilant ImageShare security is built using common security principles to ensure data security and integrity.

Default Deny

Default Deny is the concept that no unexpected input should ever be handled, except for the provision of a generic error. Default Deny ensures that there are no implicit backdoors in the system.

Enumerate Goodness

A corollary to Default Deny, Enumerating Goodness indicates that it is easier and more secure to maintain an exhaustive list of correct actions than it is to maintain an exhaustive list of threats. Moreover, it is easier and more secure to provision privilege for proper actions than to deny privilege for improper actions. Enumerating Goodness ensures that there are a limited number of ways that the system may operate, preventing even accidental misuse of the system.

Least Privilege

Least Privilege is a widely-used standard in data administration. It is, quite simply, the principle that no actor in a system should be given access to more data than is necessary for proper operation of the system. ImageShare's Application Layer provides the bulk of Least Privilege enforcement by acting as the gateway to and sole decrypting agent of the information stored in the Data Layer.

Vigilant applies this principle to the application development process to guarantee that no process within the Application Layer is given access to more data than is necessary to perform the required processing. For instance, the user-authentication system must have access to the database tables containing user login information, but does not require access to the patient-data store. Therefore, the operation of user-authentication does not have permission to access any patient data. Similarly, the application process which renders image previews does not require access to the user authentication store and is therefore unable to access that information.

It is the responsibility of the Information Technology Administrator of the client institution to ensure that their users are given least-privilege permissions. The ImageShare user administration system offers role-based control of user access. The users must be assigned to a role appropriate for their need to access information.

Least Information

A natural extension of Least Privilege, the principle of Least Information is implemented in Vigilant ImageShare to indicate that as little data as possible should be provided for a given request. For instance, a request for a patient's age does not warrant a response object that also includes a patient name and patient birth date. Moreover, this principle indicates that information which is part of the request need not be echoed in the response. Since requests for a patient's age necessarily requires the request to include a patient identifier (typically obfuscated by our data keys), it is important that the request not be unnecessarily echoed as part of the response to ensure that the patient identifier and a piece of Protected Health Information (PHI) is not unnecessarily returned in the same response object.

Risk Assessment by Layer

Physical Layer Risks

Physical risks arise only when a threat arises to the hardware acting as the server. In a traditional hosting scenario, there would be a well-defined piece of hardware housed in an office or data center. ImageShare utilizes a modern virtualized server model provided by a third party, Amazon Web Services [AWS]. AWS secures the virtualizing environment, as well as the hardware processing and storing the virtualization. AWS provides a security policy and auditing documents on their website[13].

Physical access, here, relates directly to datacenter access. ImageShare virtualized instances are hosted in the EC2 – EAST datacenter in Virginia. AWS strictly restricts access to the physical hardware in their data centers. AWS defines their datacenter physical security policy in their security documents[14]. In addition to security from unauthorized access, AWS provides extensive protections from environmental disasters, to provide uninterrupted data integrity at or beyond the industry standard for datacenter protections[15]

In this case, then, physical access also refers to the ability to access the virtualized instance in the Elastic Compute Cloud [EC2]. This access is restricted by a 2048-bit RSA key pair with the private certifying key maintained on the AWS server and public access keys are generated and assigned only to select members of the Vigilant Development and Support staff. These keys are managed and may be revoked at any time, should one become compromised. AWS strictly restricts their own employees' access to virtualized instances as described in their security document[16].

Data Layer Risks

The system Data Layer refers to the persistent storage of the virtualized server instance [VSI] as well as the run-time state of the random access memory [RAM]. Persistent storage is a combination of the file storage (the actual file contents) and the database (metadata and file system references). As data is accessed by the application layer, it is referenced in the database, and portions of the files are moved into RAM. The VSI, not AWS, is responsible for maintaining the integrity and security of the VSI's file system and database.

In a typical scenario, a patient's imaging data is uploaded to the VSI. Disregarding, for now, the protections afforded by the Client, Transport, and Application Layers, data is temporarily buffered in RAM and then recorded in two places within the Data Layer. File contents (in this case, imaging data) is stored to the file system and associated reference metadata is entered into the database. Database and file system access are thusly restricted to the application logic

[13] Amazon Web Services Security: http://aws.amazon.com/security

[14] Amazon Web Services: Overview of Security Processes, August 2010, Amazon Web Services: http://media.amazonwebservices.com/pdf/AWS_Security_Whitepaper.pdf, p.6

[15] Ibid, p.7

[16] Ibid, p.5-6

(required for the application to operate) and to the highest-ranking Vigilant system administrator assigned to maintenance on this VSI (required for maintenance and support). Access is restricted for all other purposes. No other VSI system user is authorized to view or edit objects in the Data Layer.

The Data Layer is further protected by strong 256-bit AES encryption. This encryption is provided by the Application Layer to further obfuscate data. The database is encrypted with similar technology provided by a different part of the Application Layer (SQL2008 instead of WS2008). In other words, the Data Layer merely stores and retrieves data in an encrypted format. The Data Layer itself is unable to decrypt the data it manages. In this way, direct access to the data stores does not compromise the security of its contents. That is, the Data Layer is immediately rendered useless unless access and control of the Application Layer is also available.

Application Layer Risks

The security of the Application Layer is directly dependent upon the security of the VSI's operating system. The ImageShare VSI utilizes Microsoft Windows Server 2008: Datacenter Edition [WS2008], a highly secure and well-maintained commercial server operating system. Microsoft provides extensive security documentation of the software on their website[17].

WS2008 provides an additional level of security by role-based authentication. This augments the 2048-bit RSA key pair with a high-entropy password, offering dual-layer security that first verifies the connecting machine and then the connecting user. WS2008 handles the file system utilized by the Data Layer and stores file contents using the common and high-data-integrity New Technology File System [NTFS]. WS2008 and NTFS are commonly used, enterprise-grade, server and data systems.

Vigilant has configured the ImageShare VSI's WS2008 software for a "Default Deny" approach to user-permissions. That is, by default, users are restricted from all activity, unless a particular activity is excepted for that user. The Application Layer, for instance, must have read/write access to the file system and database in order to serve data to authorized users. The Application Layer is authorized to do this as well as execute code on the VSI processor. These are the only two actions that the Application Layer is required to perform and, as such, are the only two actions permitted. The Application Layer cannot, for example, create, modify, or query system information, cannot add, drop, or modify database tables, and cannot trigger any VSI system actions such as shutdowns or restarts.

Transport Layer Risks

The Transport Layer is secured by an Equifax certificate to validate the server identification. As with any web application, it is the responsibility of the user to verify that the server is validly identified. Most modern browsers will issue an alert to the user when an inaccurate identity certificate is encountered.

[17] Secure Windows Server, Microsoft: http://technet.microsoft.com/en-us/library/dd548350%28WS.10%29.aspx

Once the server has been identified, the transport layer is further protected by the establishment of a Secure Socket Layer [SSL][18] connection with high-grade 128-bit AES encryption. It is imperative to note that this channel-level encryption is in addition to the 256-bit AES encryption active on the imaging data packets themselves. The SSL connection is established before any application data is exchanged between client and server, so no authorization information may be intercepted.

Users may verify the encryption level and server identification information through their browser's security panel.

Client Layer Risks

The Client Layer typically is the most difficult to secure. Web applications offer unique challenges in securing the Client Layer. Web applications, by their nature, are available on a variety of platforms with varying degrees of client protections and vulnerabilities: outdated browsers still in use; compromised client systems logging keystrokes; and unclosed sessions can allow access to unauthorized users.

Vigilant mitigates as many of these risks as possible, but the vast majority of viable Client Layer protections must be taken by the individual users and enforced, where possible, by the Information Technology policies of the subscribing institution.

Vigilant attempts to enforce client security by implementing as many server-side controls as possible. For instance, users of an obsolete browser (e.g. Internet Explorer 6, Mozilla Firefox 2.x) are provided a reduced feature set while using the service. This helps to limit the information displayed in a potentially unsafe browser. However, since the server depends upon the client browser to identify itself, a wholly compromised browser could spoof the user-agent information in the request.

The key to strong Client Layer security is a complete and well enforced institutional information security policy. Vigilant's mitigation schemes for the Client Layer can be compromised (as can the schemes of nearly any software system) by insecure user activity. Users should choose, or be assigned, a high entropy password. Users should actively log out of the service when not using it. Although ImageShare will automatically log out idle sessions, users should not depend on this feature as the standard log out mechanism. High entropy passwords can be enforced by regular expression on the request of the institutional administrator.

URL obfuscation is activity employed as a client layer protection. Through this, URL patterns are not easily determined and more importantly, the URL is not rewritten as a method of information persistence during a session. Although the Application Layer actively prevents unauthorized data access, URL obfuscation provides an extra layer of passive security at the Client Layer.

[18] The SSL Protocol: Version 3.0, Mozilla (Formerly Netscape Communications): http://www.mozilla.org/projects/security/pki/nss/ssl/draft302.txt

Vigilant will work with the Information Technology administrator of the subscribing institution to devise ideal Client Layer security protocols for using the ImageShare service. Similarly, extraordinary user security rules of the subscribing institution (e.g. password entropy requirements, session timeout limits), that must be implemented server-side, can be implemented by Vigilant with due notice.

Typical System Risks and Responses

Typical system risks are determined from the most recent Open Web Application Security Project [OWASP] assessment of the most common web application vulnerabilities[19]. For in-depth descriptions of the misuse scenarios, see the footnotes.

Code Injection[20]

Code injection is prevented at the Application Layer. In particular, the application must not accept code snippets as non-parameterized arguments to page templates. Vigilant ImageShare utilizes the Microsoft ASP .NET MVC2 template framework. The framework, as part of the mature ASP.NET family, has strong built-in protections against code injection.

Vigilant ImageShare follows the best practices guidance published by the Microsoft ASP.NET team regarding protection of code injection[21]. Attempts to submit SQL queries as part of a site submission are immediately thwarted by the strongly typed data-models used in the MVC2 framework. All form submissions are mapped onto strongly typed variables and will generate run-time exceptions for any attempt to submit data other than the expected type. This situation very clearly follows the Default Deny principle by rejecting anything except expected inputs.

Cross-Site Scripting[22]

Cross-Site scripting can occur on any user input that will later be displayed to other users. All form values submitted through the ASP.NET MVC2 framework are parsed for safety. Any potentially unsafe value submitted to the site is logged along with the source IP address and an alert email is sent to Vigilant developers. Although it is possible to accidentally attempt to submit unsafe HTML/JavaScript via a copy/paste command, it is Vigilant's policy to block IP addresses that repeatedly attempt to submit unsafe values to the site as this constitutes a potentially dangerous threat, whether the submission is advertent or not.

Vigilant ImageShare follows the best practices guidance published by the Microsoft ASP.NET team regarding protection from cross site scripting[23].

[19] Open Web Application Security Project: http://www.owasp.org/index.php/Main_Page
[20] OWASP: Injection: http://www.owasp.org/index.php/Top_10_2010-A1-Injection
[21] Basic Security Practices for Web Applications, Microsoft: http://msdn.microsoft.com/en-us/library/zdh19h94.aspx
[22] OWASP: Cross-Site Scripting (XSS): http://www.owasp.org/index.php/Top_10_2010-A2-Cross-Site_Scripting_(XSS)
[23] Script Exploits Overview, Microsoft: http://msdn.microsoft.com/en-us/library/w1sw53ds.aspx

Authentication and Session Management[24]

The OWASP assessment enumerates 7 questions to determine a web application's vulnerability to authentication and session exploits:

1. Are credentials always protected when stored using hashing or encryption?
   Yes, email addresses act as system usernames and are therefore stored directly in the database. Passwords, however, are never stored in a readable way. Instead passwords are hashed and the hash is stored. When a password query arrives, the query is hashed using the same certificate and only the hashes are compared. The user's true password, once hashed, cannot be recovered.
2. Can credentials be guessed or overwritten through weak account management functions (e.g., account creation, change password, recover password, weak session IDs)?
   No, account creation is protected by role as accessible only to administrators. Even a correctly guessed URL pattern to access the account creation page will fail and require authorization of an administrator-privileged user. Change password is similarly restricted by role. In the standard user role, no account modifications are permitted on other users by Application Layer authorization requirements. Recover password is unavailable due to the inability to decrypt the password hash for any user. Session IDs are 128-bit Globally Unique Identifiers [GUIDs] and are cycled at the end of every session. The probability of randomly guessing an active session ID is extraordinarily small[25]
3. Are session IDs exposed in the URL (e.g., URL rewriting)?
   No, this is easily seen by logging in to the site and reviewing the URL as an authenticated user.
4. Are session IDs vulnerable to session fixation attacks?
   No, ImageShare utilizes a double-layer session identification system where the session cookie stores both the Session ID [SID] and a hash of the username. In addition, SIDs cannot be set by the client using query string syntax in the URL. When authenticating a session, not only must the SID be active, but the cookie must also contain a valid hash of an active user account. The username hash Is generated by a salt stored securely on the server.
5. Do session IDs timeout and can users log out?
   Yes, both features are explained elsewhere in this document.
6. Are session IDs rotated after successful login?
   Yes, new SIDs are generated at login for all users. When a user arrives at the site presenting a cookie with an invalid SID, the user is directed to the authentication page, and on successful authentication, is issued a new SID to place in the cookie.

[24] OWASP: Broken Authentication and Session Management: http://www.owasp.org/index.php/Top_10_2010-A3-Broken_Authentication_and_Session_Management

[25] 2.5.5 Globally Unique Identifiers (GUIDs), Microsoft: http://msdn.microsoft.com/en-us/library/cc246025%28v=PROT.13%29.aspx

7. Are passwords, session IDs, and other credentials sent only over TLS connections?
   Yes, see the Transport Layer risk assessment above for additional information on how a TLS/SSL secure connection is established before sending any sensitive information between the server and client.

Direct Object References[26]

The ASP.NET MVC2 Framework does not make direct database queries. By utilizing an Object Relational Mapping [ORM], the ImageShare application must make all database queries on the Data Layer through a Microsoft-supplied code library. As all incoming data is checked for safety on submission, it is similarly impossible to pass raw SQL statements directly into the ORM without generating an error.

This Default Deny behavior only allows queries that contain strongly typed parameters and statements generated at compile-time. Run-time generation of complete SQL statements is prevented by the ORM.

[26] OWASP: Insecure Direct Object References: http://www.owasp.org/index.php/Top_10_2010-A4-Insecure_Direct_Object_References